CHORD: Distributed Data-sharing via Hybrid ROS 1 and 2 for Multi-robot Exploration of Large-scale Complex Environments

Muhammad Fadhil Ginting[1], Kyohei Otsu[1], Jeffrey A. Edlund[1], Jay Gao[1], and Ali-akbar Agha-mohammadi[1]

***Abstract*—A well-structured and reliable networking is key to the successful operations of *autonomous* multi-robot systems. In this paper, we present our design and implementation of a multi-robot networking architecture CHORD (Collaborative High-bandwidth Operations with Radio Droppables) based on two popular robotics middleware, ROS 1 and ROS 2. We discuss the benefit and best practices of combining two different frameworks that share the same spirit and show its performance from large-scale real-world experiments. The proposed system is developed as part of Team CoSTAR's effort for the DARPA Subterranean (SubT) Challenge.[2] The system has been field-proved and demonstrated in the Urban Circuit event, where team CoSTAR won first place. To our knowledge, this work is the first real-world demonstration of a ROS 2-based multi-robot system in such large-scale extreme environments. From the significant improvement of the communication performance and the ease of transition from existing ROS 1 systems, this work encourages wider adoption of ROS 2 in field robotics applications.**

***Index Terms*—Networked Robots, Multi-Robot Systems, Field Robots.**

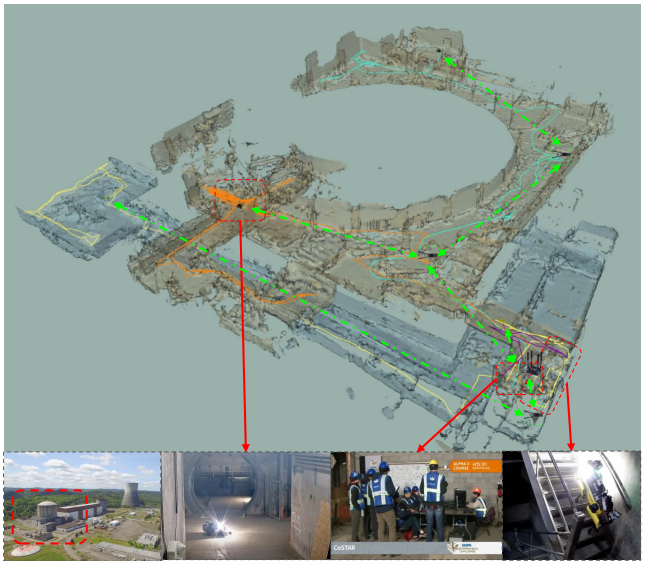

Fig. 1: This picture shows a team of four autonomous robots empowered by our CHORD networking system exploring a power plant in the DARPA Subterranean Challenge. The robot team autonomously builds the map shown in the figure. The photos are from DARPAtv [7].

I. Introduction

AUTONOMOUS multi-robot teams offer effective solutions to various robotic applications that require the exploration of large-scale complex environments. For example, a team of mobile robots aids search and rescue operations in contaminated buildings or collapsed tunnels [1], surveillance tasks [2], natural resource monitoring [3], and planetary surface/subsurface mapping [4], [5]. Communication is one of the critical challenges in enabling collaborative autonomous behaviors using robot teams in these domains. No external infrastructure is likely available, and thus robots are required to form a mobile ad-hoc network (MANET) using range-limited wireless devices. A well-structured and reliable networking architecture is essential to support the distributed system operation in these challenging environments.

In this work, we consider a supervised autonomous robot-team deployment for exploration and mapping of large-scale environments [6]. We assume a system of multiple capable robots generating megabytes of environmental data per minute. The networking system supports the robot team to carry out autonomous collaborative exploration, and at the same time, provide the opportunity for the human supervisor to oversee and interact with robots, when needed and when a communication link to the supervisor is established. This operational scenario poses requirements for the communication systems, such as reliable message transfer over dynamic intermittent networks, and real-time end-to-end communication with sufficient bandwidth for visual feedback.

This paper presents the system design and initial results of our multi-robot networking architecture named CHORD (Collaborative High-bandwidth Operations with Radio Droppables). The goal of CHORD is to maintain high-bandwidth links to multiple robots for efficient commanding and data gathering. The robots and the base station communicate over the dynamic network composed of mobile and static wireless communication nodes. We develop a hybrid of ROS 1 and ROS 2 for the communication middleware to leverage the existing code base and powerful quality of service (QoS) features for inter-robot communications. CHORD is used as Team CoSTAR's networking system at the DARPA Subterranean

Manuscript received: October 13, 2020; Revised January 9, 2021; Accepted February 3, 2021.

This paper was recommended for publication by Editor M. Ani Hsieh upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. *(Corresponding author: Muhammad Fadhil Ginting.)*

[1]All authors are with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109, USA {firstname.lastname}@jpl.nasa.gov

[2]CoSTAR Team Website: https://costar.jpl.nasa.gov

Digital Object Identifier (DOI): see top of this page.

(SubT) Challenge [7], where a robot team rapidly explores and maps the complex underground environments such as mines, subway networks, and caves. CHORD is a component of the NeBula autonomy framework [8], which was deployed onto a team of four robots during the Urban Circuit event in February 2020 (Fig. 1). The system reliably supported the team's collaborative exploration, contributing to the first-place finish in the competition.

II. Related Works

Multi-agent communication systems have been considered in various domains in robotics, including cooperative exploration [6], collaborative SLAM [9], [10], and ad-hoc communication infrastructure building [11]. However, it is still challenging to maintain a reliable system under harsh real-world constraints. Efforts have been made to tackle similar real-world challenges in the context of robot tele-operation on disaster response [1] and robot team exploration of tunnel-like environments [12], [13]. Despite the growth and increasing attention in this field, there are still open questions and engineering challenges to build a reliable mobile network without external communication infrastructure, especially in large-scale environments.

Robotics frameworks. Various robotics frameworks and middleware are available to foster robotics application development. Many frameworks can be extended to the multi-robot communication system such as Player/Stage project [14], Orca [15], Urbi [16], ROS [17], RT Middleware [18], and MIRO [19]. In this work, we build a system based on ROS since it has a powerful development ecosystem with a vast collection of robotics tools, libraries, and community support. ROS's communication is based on the publish/subscribe model, which provides flexibility by decoupling sender and receiver implementation. A major drawback of ROS in the multi-agent scenario is the presence of a centralized module (ROS master), making it challenging to apply the framework to distributed systems on non-ideal networks.

ROS for multi-robot systems. There are various approaches to overcome the limitation of the centralized component in ROS and extend to multi-robot systems. One popular package is `multimaster_fkie` which offers the multi-master extension to the existing ROS master system [20]. The `multimaster_fkie` provides automatic master discovery via UDP multi-/uni-cast and synchronizing their states by exchanging lists of publishers and subscribers. Another approach is to use a bridge between different ROS master networks. The bridge can be implemented in various ways depending on its networking requirements, including Data Distribution Service (DDS) [21], ZeroMQ [22], custom UDP and TCP-based protocols [23], [6].

ROS 2. The next generation of ROS, called ROS 2, is being actively developed. Unlike the original ROS (referred to as ROS 1), ROS 2 has introduced multi-robot use cases in its design [24]. The major upgrade in ROS 2 is the adoption of DDS as the underlying communication middleware, a well-proven technology that helps us build a scalable and robust communication system. DDS has a distributed discovery system by default and shares a similar publish/subscribe transport concept with ROS. DDS also offers Quality of Service (QoS) features that gives flexibility in controlling communication behavior. ROS 2 is still in an early phase of development and far behind ROS 1 in terms of the number of packages and active developers. The literature on ROS 2 communication system is limited: an early evaluation of the ROS 2 communication mechanism [25], [26] and preliminary industrial use cases [27], [28].

Contribution. In this paper, we perform a principled analysis for the development of high-bandwidth data-sharing systems, and we present a practical implementation with the hybrid of ROS 1 and 2. We assess the performance and resiliency of CHORD during multi-robot operations in the context of the DARPA SubT Challenge. To our knowledge, this is the first real-world demonstration of a ROS 2-based multi-robot system in such large-scale extreme environments. We believe our lessons learned not only benefit the ROS-based community, but will also guide the development of communication protocols for generic data-sharing systems.

III. System Design

This section discusses the requirements and system design that enables reliable multi-robot communications.

A. Requirements

Sender/receiver decoupling. The publish/subscribe paradigm provides the ability to decouple data producers and consumers, thereby increasing scalability, flexibility, and robustness. There are three dimensions for this decoupling [29]:

- *Space*: Data producers and consumers do not have to know each other.
- *Time*: Data producers and consumers do not have to both be active at the time of interaction.
- *Synchronization*: Data producers and consumers are not blocked while sending/receiving data.

The ROS 1 communication system was built on top of the publish/subscribe paradigm. However, it lacks critical features to provide these decoupling dimensions. These requirements must be fully met to support inter-robot communications over non-ideal networks.

Dynamic network support. In our application, the robots form a wireless mobile ad-hoc network. The network topology can change dynamically along with its link properties such as available bandwidth, delay, and data loss rate. Our communication framework must support the dynamic nature of the network. It is especially important that the communication does not depend on a central component, which can be easily broken when multiple robots explore an unknown communication-limited environment. In addition, advanced QoS features can be used to guarantee delivery of important priority data while maintaining network stability over low-bandwidth links. ROS 1 is limited in both: it depends on a centralized daemon (roscore) to establish connections and provides a limited number of QoS parameters (reliability, buffer size, 'latched' delivery).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2021.3061393, IEEE Robotics and Automation Letters

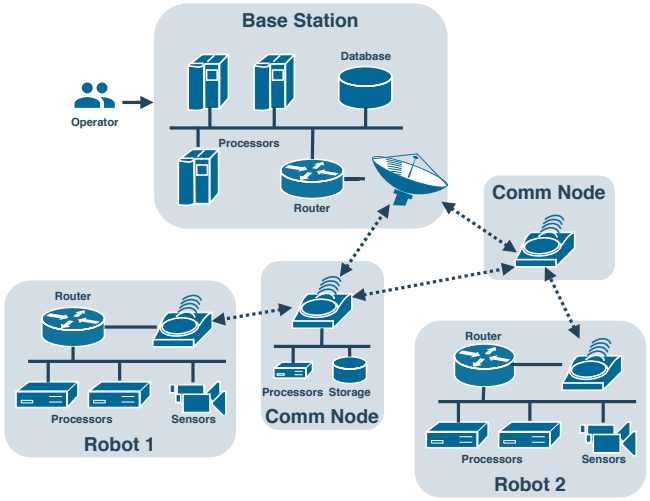

Fig. 2: CHORD network architecture

ROS 1 compatibiliy. Although ROS 1 has limitations in multi-robot communications, it is an excellent software framework that allows easy development of robotics applications and has strong community support. There are also many useful libraries developed using the ROS 1 framework, including our in-house software suites. Eventually, the entire community is expected to shift to the ROS 2 framework. However, at the time of writing, it is still reasonable to develop applications with ROS 1. With this heterogeneous system, the communication system needs to provide transparency to the users; i.e., the system should provide the users the ability to develop their application using the ROS 1 messaging framework without knowing the details of the underlying communication system.

B. Architecture Design

We develop a multi-robot networking system named CHORD. The architecture diagram is shown in Fig. 2. We assume three types of agents: 1) static agents (e.g., base station), 2) mobile agents (e.g., robots), and 3) semi-mobile agents (e.g., communication relay nodes). Semi-mobile agents cannot move on their own but are small enough to be carried and deployed from mobile agents. Each agent has its private network that connects one or more processors and sensors. These agents are connected via a wireless mesh network.

We use ROS 1 for intra-robot communications, and use ROS 2 for inter-robot communications. Each robot has a ROS 1/2 bridge that translates the message format between the two. Fig. 3 and Fig. 4 depict the difference between intra- and inter-robot communications. Inside a robot's ROS 1 network, node-to-node communication is done with ROS 1's master-based handshaking. For ROS 1 nodes running on different robots, the messages are first delivered to the ROS 1/2 bridge node (which is a ROS 1 node as well as a ROS 2 node), transferred to the other side of the bridge via the ROS 2 (essentially DDS) protocol, and converted back to the ROS 1 message. For ROS 2 topics (i.e., inter-robot topics), we configure QoS by using DDS parameters. The powerful QoS control mechanism of DDS enables traffic prioritization and resource control with ease. All the data that goes into the inter-robot network are

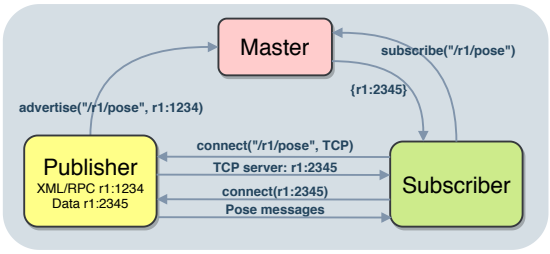

Fig. 3: Intra-robot messaging process with ROS 1 publish/subscribe scheme (figure adapted from ros.org). Note that "r1:1234" represents the port 1234 on host r1.

throttled and compressed to reduce the network load. We also performed algorithmic data fragmentation if possible. For example, a robot incrementally sends part of the map to be later reassembled to a full map on the base station.

In the proposed system, the inter-robot communication is fully transparent to the users. Therefore, it is straightforward to adopt advanced routing technologies without changes in the application code. Examples include a data mule, that carries other robot's data on its behalf, and a static local data store, that temporarily keeps robot's data until another nearby robot drives by to collect.

IV. ROS 1/2 Hybrid System

This section describes the implementation details of the ROS 1 and ROS 2 hybrid communication framework.

A. ROS 1 internal communications

Our robots have multiple onboard computers, connected to the same network via Gigabit Ethernet. The main computer manages the ROS 1 master processes, and all the ROS 1 nodes point to the same master. We utilize the following ROS 1's QoS control as appropriate:

- TCPROS: TCP-based message delivery (equivalent to "reliable" reliability policy in ROS 2).
- UDPROS: UDP-based message delivery (equivalent to "best-effort" reliability policy in ROS 2).
- Latching: Send the last message for late-joining subscribers (equivalent to "transient local" durability policy with a queue size of 1).

We use TCPROS for most of the topics, and enable latching for low-rate update topics requiring reliable transfer. Since all the onboard machines are connected with high-speed cables, we only use UDPROS for temporary debugging purposes where we need to stream high-volume data over wireless networks.

B. ROS 1/2 Bridge

The ROS 1/2 bridge provides a network bridge to enable message exchanges between ROS 1 and ROS 2 systems. Our ROS 1/2 bridge implementation is based on the `ros1_bridge` package[1]. A ROS 1/2 bridge node consists of a ROS 1 node and a ROS 2 node that pass the data between the two ROS worlds. An agent only needs one ROS 1/2 bridge

[1]https://github.com/ros2/ros1_bridge

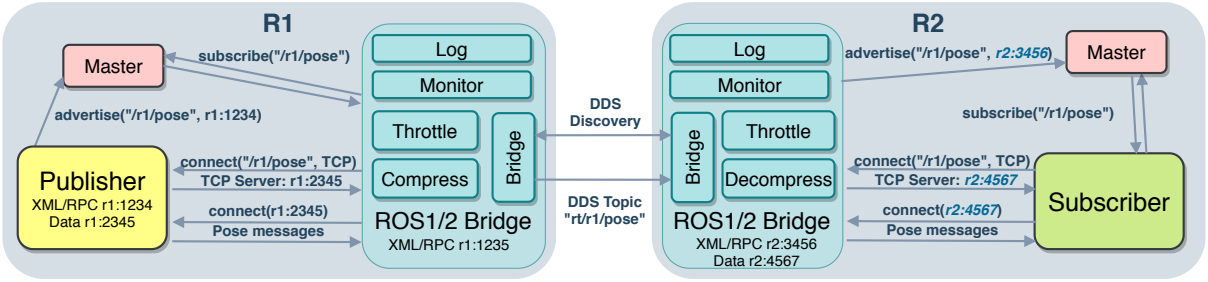

Fig. 4: Inter-robot messaging process with ROS 1/2 hybrid system.

node to transmit and receive data through the ROS 2 network. In our work, we expand the functionality of `ros1_bridge` to support topic-based QoS configuration.

We only pass subsets of ROS 1 message topics that are relevant to other agents to the network bridge. All the data sent to the inter-robot network is throttled and compressed on the bridge. We apply a rate-based throttling and bandwidth-based throttling. We use a bzip2-based lossless compression algorithm for all the topics by leveraging the existing `blob` package[2]. Higher-efficiency methods such as LZ4 may be chosen based on the application requirements. Additionally, JPEG compression is applied to the image topics to obtain higher compression rates.

C. QoS Configuration

A QoS configuration is essential to manage data traffic across communication networks. Depending on the mission requirement and network quality, we configure the QoS profiles for each topic in ROS 2. For more information about various QoS configuration, readers are referred to the vendor-specific documentation such as [30].

Proper QoS has to be set to maximize the overall network performance. There are two major topic classification in our system: 1) topics that require real-time transmission for situational awareness (e.g., vehicle status), and 2) topics that require full message history transfer for the post-processing or delivering the mission-critical information. For the topics with real-time requirements, we use a *best effort* reliability policy, *volatile*[3] durability policy, and *keep last* history policy with the queue size of 1. On the other hand, for the topics with reliability requirements, we use a *reliable* reliability policy, *transient local*[3] durability policy, and *keep all* history policy so that the messages are reliably delivered even though the network may be down for periods of time. Another useful QoS is the lifespan policy, with which we can control how long the system attempts to deliver the messages. The lifespan policy is set when old unused data may congest the network.

D. DDS Performance Tuning

ROS 2 supports multiple DDS vendors. We choose Fast RTPS, the default ROS Middleware Interface, as our underlying DDS system.

[2]http://wiki.ros.org/blob

[3]The *volatile* policy does not keep messages for future use. Conversely, the *transient local* policy stores messages to deliver to late-joining subscribers.

Each DDS vendor provides a way to configure its behavior at run time. We fine-tuned DDS parameters to achieve better performance on our hardware. Fast RTPS can be configured with XML profiles, which allow us to access transport descriptor, participant, publisher, and subscriber settings. In the transport descriptor profile, we defined the maximum initial peer range and the interface white list. The maximum initial peer range is the maximum number of guessed initial peers to try to connect to. We set this value to the maximum number of agents that we plan to use since larger values take a long time to initialize the DDS network. We also use the interface white list to limit the interfaces that can be used by DDS. In the participant profile, we configured the unicast locator to mitigate speed and reliability issues experienced with the multicast locator over our existing network/radios To enable the discovery with the unicast network, we specified the network IP addresses in the initial peer list. Meanwhile, in the publisher and subscriber profiles, we set up additional QoS settings and the flow controller. The flow controller or throughput controller limits the publishers' data bandwidth as a fail-safe to protect network stability.

V. Experimental Results

We conducted a series of field demonstrations of our CHORD system at various indoor and outdoor fields with different robot configurations. In this paper, we mainly report the communication performance of Team CoSTAR's autonomous multi-robot operations in a large-scale complex power plant in Elma, WA, during the DARPA SubT Challenge Urban Circuit [31]. The course covers two floors of the power plant with a size of around $90 \times 90 \times 15\ m^3$. There are many small rooms and narrow corridors divided by thick walls that prevent direct wireless communications. The mission of the robot team is to explore the course rapidly, search for artifacts, and report the locations of the artifacts to the human supervisor at the base station in 60 minutes.

A. Hardware Setup

Communication radios: The networking system uses a decentralized layer-2 mobile ad-hoc mesh network (MANET). We use commercial off-the-shelf MANET radios from Silvus Technologies (Streamcaster 4240 for the robots and 4400 for the base station). In addition to the main radio, the extra radios are loaded in each robot and can be dropped to build the backbone network [32].

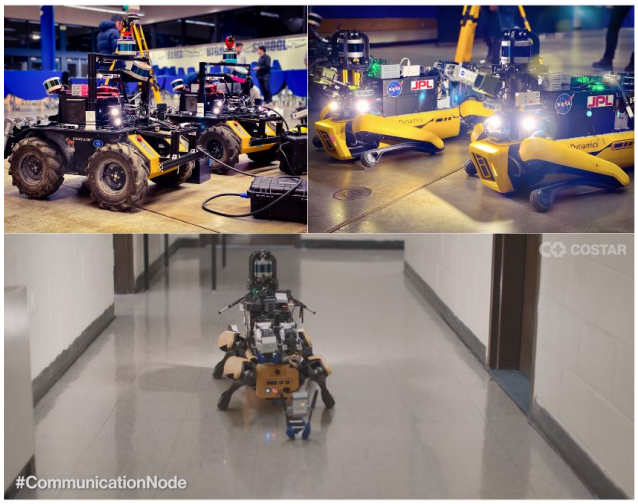

Fig. 5: CoSTAR's Robots in the DARPA SubT Challenge Urban Circuit (top). Each robot can drop communication nodes to build a backbone wireless mesh network as well as to aid robot localization using ultra-wide-band (UWB) nodes [33] (bottom).

Robots: The robots used in our experiments are two Boston Dynamics Spot quadrupeds and two Clearpath Husky-A200 series ground vehicles shown in Fig. 5. Each robot is equipped with a communication node dropper that can deploy multiple radios.

Base station: The base station consists of several computers. The main computer is responsible for communicating with the robots in the field (receives incoming telemetry data and sends commands). The received data is shared with other base station computers via Gigabit Ethernet. We also had a dedicated computer to monitor the data traffic on the main computer interfaces.

B. End-to-end Statistics

In this section, we evaluate the performance of end-to-end communication between the autonomous robot Husky 4 and the base station. During this experiment, the network topology changed from a single hop (direct) to four hops. The experiment duration was 60 minutes and the robot went outside of the communication range for several times during the autonomous navigation phase.

Timing statistics. Table I shows the statistics of major ROS topics communicated between Husky 4 and the base station. Different QoS profiles were used based on the topic classification in Section IV-C. We reported vehicle status, velocity command, and TF tree for the real-time topics and key lidar scans [34], pose graph [9], Information Roadmap (IRM) [35], [36], and artifact detection [37] for the mission-critical topics. Topics with real-time requirements achieved a nominal delay of < 50 ms for all topics. For comparison, the average UDP one-way ping time to Husky 4 was 22.9 ms before the run started and 37.8 ms at the end of the run. The message delivery ratio (MDR) for real-time topics was lower than 100% since message delivery is not guaranteed with this category. For example, messages are dropped if there is no immediate communication route between two entities. Reliable topics, on the other hand, achieved 100% MDR with a large maximum delay. The large maximum delay of reliable topics shows the ability of the ROS 2 communication layer to automatically cache data until it is received by the ROS 2 subscribers on the other entity.

TABLE I: End-to-end ROS topic statistics. Topics are categorized into two groups based on the classification in IV-C. MDR stands for Message Delivery Ratio.

Topic	Average Size (kB)	Median/Max Delay (second)	MDR (%)
1) Real-time topics			
Vehicle Status	0.25	0.042 / 2.31	92.48
Velocity Command	0.06	0.020 / 0.31	98.91
TF tree	0.35	0.036 / 1.85	92.56
2) Reliable (History-dependent/mission-critical) topics			
Key Lidar Scans	29.82	0.075 / 268.80	100.0
Pose Graph	0.34	0.047 / 269.26	100.0
Information Roadmap	0.19	0.035 / 218.64	100.0
Artifact Detection	159.06	0.170 / 235.71	100.0

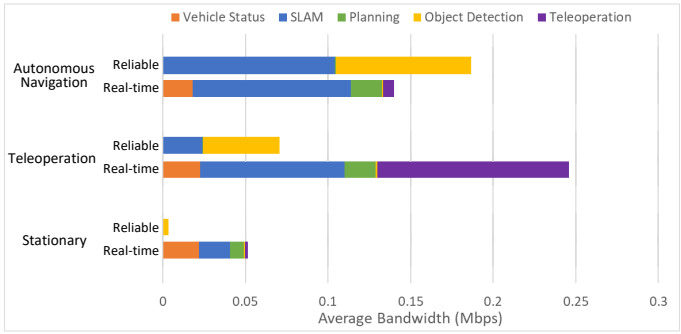

Fig. 6: Time-averaged bandwidth usage of Husky 4 for three different operation modes. The ROS topics are grouped into 5 categories based on the applications.

ROS topic bandwidth statistic. Fig. 6 summarizes the bandwidth usage averaged over time for three distinct robot operation modes: stationary, autonomous navigation, and tele-operation. In all these modes, we were able to fit inside the bandwidth budget, which contributed to the stability of the dynamic network. Although teleoperation and autonomous navigation used about the same amount of bandwidth, their data profiles were different. Teleoperation required one third of the total bandwidth to stream images from vehicle and control the robot manually from the base station. Note that almost all of the mission is autonomous, and tele-operation mode is only used when the human supervisor aims at injecting a behavior outside the mission specification. This happens very infrequently and the multi-robot team is fully autonomous during the nominal mission execution phase.

C. Multi-robot Operations

In this section, we analyze the multi-robot CHORD system and highlight aspects that affect the system performance.

Handling lossy networks. Exploring an underground power plant poses significant challenges for communication. The walls are thick and it is hard to maintain line of sight connections between moving robots, communication nodes, and the base station. Fig. 7 shows the dynamically changing network topology. As seen in the figure, the exploring robot can easily lose connection to the base station. One of the communication losses occurred with Husky 4 at $t = 22.6$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2021.3061393, IEEE Robotics and Automation Letters

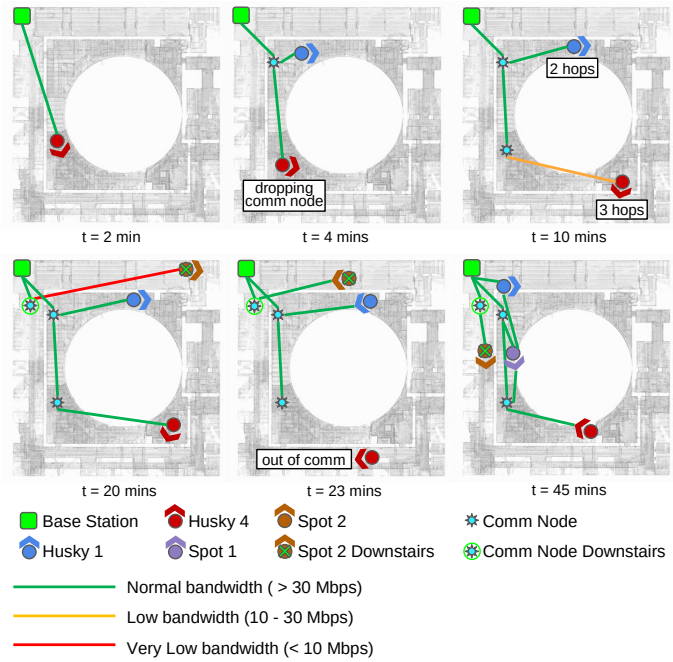

Fig. 7: Agent's location and communication topology over time from the top-view map of the 2-level course. The four robots, two Husky ground vehicles and two Spot quadrupeds, start from the base station. The robot is marked on the map when it enters the course and the communication node is marked when it is dropped.

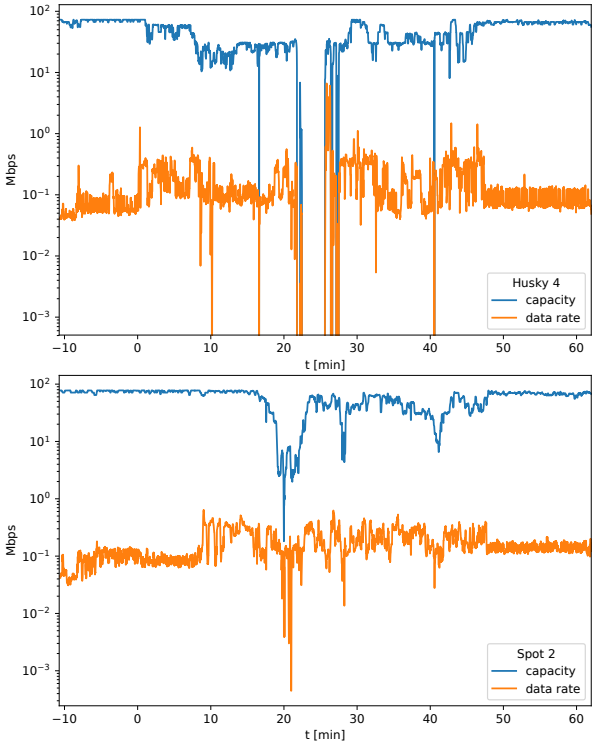

Fig. 8: Estimated network route capacity and actual data usage from Husky 4 (top) and Spot 2 (bottom) to the base station over the run time. The route capacity for both robots are dynamically changing. As illustrative examples, at $t = 22.6$ minutes, Husky 4 goes out of the mesh network; at $t = 21$ minutes Spot 2 goes to very low route capacity when autonomously exploring a room downstairs 9 meters and 80 meters horizontally away from starting gate. See locations and routes in Fig. 7. Capacity and data usage were sampled once a second from the radios using Silvus's Streamscape API. A 10 second rolling median was applied.

minutes, where the estimated network capacity between Husky 4 and the base station drops to zero (Fig. 8). When robots are outside the communication network, the base station is unable to communicate with the robot. In this example, as Husky 4 returned to the mesh network, the data flow spiked as data from the autonomously explored region was transmitted by the robot. This data burst came from messages with Reliable QoS profile since the robot continues to try to transmit the mission-critical messages until the base station receives them. After the data burst, the data rate went back to normal again, validating that the ROS 2 network can handle the lossy and unstable network.

Building communication backbone. Maintaining strong communication links between base station and autonomous robots is crucial to ensure sufficient bandwidth. To maintain high bandwidth routes, we drop communication nodes to extend the network. An example of this can be seen in Fig. 7. Initially, we sent the first robot (Husky 4) to autonomously explore the course ($t = 2$ minutes). To maintain communication to the first robot, the second autonomous robot (Husky 1) entered the course and dropped a communication node ($t = 4$ minutes). The communication node maintained a high bandwidth link between Husky 4 and the base station, which allowed Husky 1 to start exploring toward a different area.

VI. Lessons Learned

This section discusses lessons learned from our design, implementation, and deployment of the CHORD system.

A. Upgrading from ROS 1 System

We first discuss the migration effort from our previous ROS 1-based approach [6], focusing on added functionalities, development, and maintenance cost. The previous approach, which was used for a similar four-robot system during the DARPA SubT Challenge Tunnel Circuit in August 2019, used the popular `multimaster_fkie` package [20] and a custom cross-master messaging mechanism. To incorporate lessons from the first circuit event and support emerging communication requirements, we designed, implemented, and tested the new CHORD architecture between two circuit events. In this section, we discuss the pros/cons of our approach through the experience of deploying and managing two operable communication architectures on real-world systems.

Configurability. One of the most considerable benefits of ROS 2 is the powerful QoS and DDS configuration parameters. These knobs enable us to easily modify the communication behaviors per topic and per machine. With ROS 1, the same features need to be implemented with additional scripting. For example, we needed to store critical data on robots while the robot is out of communication range and send the buffered data when the robot rejoins the mesh network. This feature requires an additional node to store, book-keep, and resend messages based on the connectivity status. Furthermore, we need to throttle data traffic when rejoining the network to not overwhelm the link, which can become complex if there are multiple topics to be managed. With ROS 2, both behaviors can be implemented by using standard DDS features. The well-supported DDS features allow us to quickly adapt to

application requirements and save time to develop/maintain communication-related code.

Network Isolation. In our previous system, we experienced occasional network instability due to excessive bandwidth usage. The main cause was unintended data flows caused by the misconfiguration of multimaster synchronization. In the ROS 1-based system, it needs careful attention to clearly separate data sharing between two ROS networks because some standard ROS functionalities such as TF or diagnostics rely on topics in the global namespace, which can cause name conflicts. By using a different protocol for inter-robot communication, we can isolate the ROS 1 networks completely and avoid unintended data flows between agents. The network isolation also helps to diagnose networking issues easily as every inter-robot ROS topic passes through the bridge node. A downside for the bridge approach is extra complexity added to the system. If the inter-robot communication is not complex, the development cost might exceed the benefit of network isolation.

Ease of Migration. Among other competing protocols, ROS 2 has a strong advantage in migrating from ROS 1 systems. There are many concepts and terminology in common. ROS 2 provides compatibility to its predecessor by providing the off-the-shelf `ros1_bridge` package and inheriting standard message types. The ease of migration contributed a lot to developing the new architecture within a short time frame. However, it should also be noted that there are entirely new concepts in ROS 2, such as DDS, which requires its own consideration. Developers might need some experimentation since DDS features are dependent on the underlying network (e.g., discovery).

Performance. The ROS 2-based CHORD system had better end-to-end communication performance than our previous ROS 1 system. We analyzed the custom multi-master mechanism since the common topics were transmitted on our customized protocol. We observed the nominal delay of periodic real-time topics in ROS 2 (< 0.05 second as shown in Table I) is much lower than similar topics transmitted with our ROS 1-based multi-master system, which has the nominal delay of 1.5 second. The higher value of the nominal delay is mainly due to the network instability and delay introduced by the custom message conversion and transmission mechanism. Moreover, the reliability of message delivery was better on the CHORD system. In the experiment reported in Table I, the Message Delivery Ratio (MDR) for a periodically-published topic was 92.5%. This corresponds to the time that the robot was in the communication range (92.5% of the time). For ROS 1 system, the MDR of a similar topic was 82.1%, which was slightly lower than the time the robot was in the communication range (83.2% of the time), indicating unintended message loss.

B. Generic Design Considerations

Beyond the hybrid ROS 1 and ROS 2 specific implementation, there are some key lessons that apply to generic communication systems design. This section presents such generic design considerations derived from the development of multi-robot communication architecture.

Mixing Requirements. One key lesson is to allow for multiple (sometimes conflicting) communication requirements within a system. In our application, we had different requirements for intra-robot communications (e.g., efficiency, low latency, ease of algorithm development) and inter-robot communications (e.g., limited bandwidth budget, flexibility, intermittent connection). Using multiple protocols with bridges is a viable solution if there is no existing middleware that satisfies all the requirements. To make inter-operation possible, it would be preferable for the frameworks to provide a bridge to other protocols (similar to ROS 1 and 2 bridge).

Middleware Selection. When selecting a communication middleware, certain features significantly simplify development and diagnosis tasks. *Network monitoring* is one such feature when building a complex system. Hence, existence of a built-in monitoring tool is a key consideration. Monitoring can be either centralized (e.g., monitoring on bridges) or distributed (e.g., ROS topic statistics) depending on the system architecture. To handle lossy low-bandwidth networks, *QoS control* is essential. The QoS helps prioritizing certain messages over others when the network condition is not ideal. *Bandwidth management* is a related important feature when operating on a non-ideal network. The ability to manage bandwidth in different granularity levels (e.g., per topic, process, machine) is useful for fine-tuning the system and keeping the network stable.

C. Scalability Challenges

It is important to consider scalability when designing a communication architecture. The CHORD system was designed based on Team CoSTAR's system assumption, which is a network of roughly 30 nodes (at the time of submission), consists of < 10 mobile robots and < 20 static nodes. The first consideration is the scalability of the underlying MANET. The selected device and protocol should handle the target network size in the nominal operating conditions. The initial DDS discovery takes time, especially if there are many participants. This issue can be mitigated by using the static unicast-based configuration. From the data transmission perspective, it is essential to quantify the nominal available bandwidth of a wireless link and determine the bandwidth allocation that does not exceed the capacity of the bottleneck link. The compression and throttling parameters need to be set to respect this allocation. Additional per-machine traffic shaping helps to suppress peak throughput and keep the network stable.

VII. Conclusions

We presented our approach to multi-robot networking using ROS 1 and ROS 2 as underlying communication middleware. The system design and implementation detail are intensively tested within autonomous four-robot operations including the DARPA SubT Challenge Urban Circuit. Based on the lessons from the deployment into large complex environments, we identified key considerations to design a high-bandwidth data-sharing system over a lossy dynamic network composed of static and dynamic nodes. From its compelling results in large-scale experiments, this work encourages wider adoption of

ROS 2 in field robotics applications, with the hope of opening possibilities to easily develop durable multi-robot systems.

ACKNOWLEDGMENT

The research was partially carried out at the Jet Propulsion Laboratory, California Institute of Technology. This work was partially funded by the Defense Advanced Research Projects Agency (DARPA). We gratefully acknowledge William Walsh, Gregory Miles, Carlyn Ann Lee, and Gustavo Correa for the useful discussions and contributions to the initial work of our communication system.

REFERENCES

[1] K. Nagatani, S. Kiribayashi, Y. Okada, K. Otake, K. Yoshida, S. Tadokoro, T. Nishimura, T. Yoshida, E. Koyanagi, M. Fukushima, and S. Kawatsuma, "Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots," *Journal of Field Robotics*, vol. 30, no. 1, pp. 44–63, 2013.

[2] N. Michael, E. Stump, and K. Mohta, "Persistent surveillance with a team of MAVs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 2708–2714.

[3] J. J. Roldán, P. G. Auñón, M. Garzón, J. de León, J. del Cerro, and A. Barrientos, "Heterogeneous multi-robot system for mapping environmental variables of greenhouses," *Sensors*, vol. 16, p. 1018, 2016.

[4] A. Agha, K. L. Mitchell, and P. J. Boston, "Robotic Exploration of Planetary Subsurface Voids in Search for Life," in *AGU Fall Meeting Abstracts*, 2019.

[5] M. F. Ginting, T. Touma, J. A. Edlund, A. Buscicchio, , K. Otsu, and A. Agha, "Deployable Mesh Network for Enabling Reliable Communication from within Subsurface Voids to the Planetary Surface," in *AGU Fall Meeting*, 2020.

[6] K. Otsu, S. Tepsuporn, R. Thakker, T. S. Vaquero, J. A. Edlund, W. Walsh, G. Miles, T. Heywood, M. T. Wolf, and A. Agha-mohammadi, "Supervised autonomy for communication-degraded subterranean exploration by a robot team," in *IEEE Aerospace Conference*, 2020, pp. 1–9.

[7] DARPA Subterranean Challenge. [Online]. Available: https://www.subtchallenge.com

[8] A. Agha-mohammadi, et al., "NeBula: Quest for robotic autonomy in challenging environments; TEAM CoSTAR at the DARPA subterranean challenge," *(Under Review)*, 2021.

[9] K. Ebadi, Y. Chang, M. Palieri, A. Stephens, A. Hatteland, E. Heiden, A. Thakur, B. Morrell, L. Carlone, and A. Agha-mohammadi, "LAMP: Large-scale autonomous mapping and positioning for exploration of perceptually-degraded subterranean environments," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 80–86.

[10] P.-Y. Lajoie, B. Ramtoula, Y. Chang, L. Carlone, and G. Beltrame, "DOOR-SLAM: Distributed, online, and outlier resilient SLAM for robotic teams," arXiv:1909.12198, 2019.

[11] O. Cetin and I. Zagli, "Continuous airborne communication relay approach using unmanned aerial vehicles," *Journal of Intelligent and Robotic Systems*, vol. 65, no. 1-4, pp. 549–562, 2012.

[12] D. Tardioli, D. Sicignano, L. Riazuelo, A. Romeo, J. L. Villarroel, and L. Montano, "Robot teams for intervention in confined and structured environments," *Journal of Field Robotics*, vol. 33, no. 6, pp. 765–801, 2016.

[13] D. Tardioli, L. Riazuelo, D. Sicignano, C. Rizzo, F. Lera, J. L. Villarroel, and L. Montano, "Ground robotics in tunnels: Keys and lessons learned after 10 years of research and experiments," *Journal of Field Robotics*, vol. 36, pp. 1074–1101, 2019.

[14] B. Gerkey, R. Vaughan, and A. Howard, "The Player/Stage project: Tools for multi-robot and distributed sensor systems," *International Conference on Advanced Robotics*, 2003.

[15] A. Brooks, T. Kaupp, A. Makarenko, S. Williams, and A. Orebck, "Orca: A component model and repository," *Software Engineering for Experimental Robotics. Springer Tracts in Advanced Robotics*, vol. 30, pp. 231–251, 2007.

[16] J.-C. Baillie, A. Demaille, Q. Hocquet, M. Nottale, and S. Tardieu, "The Urbi universal platform for robotics," *International Workshop on Standards and Common Platform for Robotics*, 2008.

[17] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system," vol. 3, 2009.

[18] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and W.-K. Yoon, "RT-middleware: Distributed component middleware for RT (robot technology)," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 3933 – 3938.

[19] H. Utz, S. Sablatng, S. Enderle, and G. Kraetzschmar, "Miro - middleware for mobile robot applications." *IEEE Transactions on Robotics*, vol. 18, pp. 493–497, 2002.

[20] A. Tiderko, F. Hoeller, and T. Rhling, "The ROS multimaster extension for simplified deployment of multi-robot systems," *Studies in Computational Intelligence*, vol. 625, pp. 629–650, 2016.

[21] R. Hartanto and M. Eich, "Reliable, cloud-based communication for multi-robot systems," in *IEEE International Conference on Technologies for Practical Robot Applications*, 2014, pp. 1–8.

[22] ZeroMQ-ROS. [Online]. Available: https://github.com/wallarelvo/zmqros

[23] M. Schwarz, T. Rodehutskors, D. Droeschel, M. Beul, M. Schreiber, N. Araslanov, I. Ivanov, C. Lenz, J. Razlaw, S. Schüller, D. Schwarz, A. Topalidou-Kyniazopoulou, and S. Behnke, "NimbRo Rescue: Solving Disaster-response Tasks with the Mobile Manipulation Robot Momaro," *Journal of Field Robotics*, vol. 34, no. 2, pp. 400–425, 2017.

[24] ROS2 Design–Why ROS 2? [Online]. Available: https://design.ros2.org/articles/why_ros2.html

[25] Y. Maruyama, S. Kato, and T. Azumi, "Exploring the performance of ROS2," in *International Conference on Embedded Software*, 2016, pp. 1–10.

[26] C. S. V. Gutiérrez, L. U. S. Juan, I. Z. Ugarte, and V. M. Vilches, "Towards a distributed and real-time framework for robots: Evaluation of ROS 2.0 communications for real-time robotic applications," arXiv:1809.02595, 2018.

[27] E. Erös, M. Dahl, K. Bengtsson, A. Hanna, and P. Falkman, "A ROS2 based communication architecture for control in collaborative and intelligent automation systems," arXiv:1905.09654, 2019.

[28] E. Ers, M. Dahl, A. Hanna, A. Albo, P. Falkman, and K. Bengtsson, "Integrated virtual commissioning of a ROS2-based collaborative and intelligent automation system," in *IEEE International Conference on Emerging Technologies and Factory Automation*, 2019, pp. 407–413.

[29] P. T. H. Eugster, P. A. Felber, R. Guerraoui, and A. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys*, vol. 35, no. 2, pp. 114–131, 2003.

[30] *FastRTPS Documentation*, eProsima, 4 2020, release 1.10.

[31] A. Bouman∗, M. Ginting∗, N. Alatur∗, M. Palieri, D. Fan, T. Touma, T. Pailevanian, S. Kim, K. Otsu, J. Burdick, and A. Agha-mohammadi, "Autonomous Spot:long-range autonomous exploration of extreme environments with legged locomotion," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, NV, 2020.

[32] T. S. Vaquero, M. S. da Silva, K. Otsu, M. Kaufmann, J. A. Edlund, and A. Agha-mohammadi, "Traversability-aware signal coverage planning for communication node deployment in planetary cave exploration," in *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2020.

[33] N. Funabiki, B. Morrell, J. Nash, and A. a. Agha-mohammadi, "Range-aided pose-graph-based slam: Applications of deployable ranging beacons for unknown environment exploration," *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 48–55, 2021.

[34] M. Palieri, B. Morrell, A. Thakur, K. Ebadi, J. Nash, A. Chatterjee, C. Kanellakis, L. Carlone, C. Guaragnella, and A. Agha-mohammadi, "LOCUS - A Multi-Sensor Lidar-Centric Solution for High-Precision Odometry and 3D Mapping in Real-Time," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 421–428, 2020.

[35] S.-K. Kim∗, A. Bouman∗, G. Salhotra, D. D. Fan, K. Otsu, J. Burdick, and A. Agha-mohammadi, "PLGRIM: Hierarchical value learning for large-scale exploration in unknown environments," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 31, 2021.

[36] A. Agha-mohammadi, S. Chakravorty, and N. Amato, "FIRM: Sampling-based feedback motion planning under motion uncertainty and imperfect measurements," *International Journal of Robotics Research (IJRR)*, vol. 33, no. 2, pp. 268–304, 2014.

[37] E. Terry, X. Lei, B. Morrell, S. Daftry, and A. Agha-mohammadi, "Object and gas source detection with robotic platforms in perceptually-degraded environments," in *RSS Workshop on Robots in the Wild*, 2020.